# بحث PHP

أعداد المُهندسه :

رغد حمود محمد الحُمري

د/ أبراهيم الشامي

# الإضافات والتحسينات التى تم إضافتها فى PHP 8

مقدمة:

تم إصدار **PHP 8** في 26 نوفمبر 2020 كان هذا الإصدار من PHP بمثابة تحديث رئيسي يضم العديد من التحسينات والإضافات التي تجعل لغة البرمجة أكثر قوة ومرونة. في هذا البحث، سنستعرض أبرز التحسينات والإضافات التي تم إضافتها في PHP 8 .

## 1.تحسينات فى الأداء

يعتبر **الأداء** من أبرز القضايا التي تم تحسينها في PHP 8 تم إدخال العديد من التحسينات التي تساعد في تسريع تنفيذ الأكواد وجعل لغة PHP أكثر كفاءة.

### JIT (Just In Time Compilation) :

- واحدة من أكبر الإضافات في PHP 8 هي إضافة **JIT compiler**
- **JIT**هو تقنية لتحويل الأكواد البرمجية إلى تعليمات الآلة في وقت تنفيذ البرنامج، مما يزيد من سرعة تنفيذ التطبيقات.
- تعمل **JIT**على تحسين الأداء بشكل عام، خصوصاً في العمليات الحسابية المعقدة مثل الرسومات أو التطبيقات التي تتطلب معالجة بيانات كثيفة.

### تحسينات على محرك Zend :

- محرك **Zend Engine** في PHP 8 حصل على العديد من التحسينات مثل تحسينات على الذاكرة وأداء الحسابات المعقدة.
- **PHP 8**هو أسرع من **PHP 7.4**بفضل هذه التحسينات.

## 2.الإضافات الجديدة فى PHP 8 :

### 1.(خاصية) Attributes السمات:

- **Attributes** هي ميزة جديدة تسمح للمطورين بتحديد معلومات ميتا مضافة للأنواع(classes ، functions، properties، إلخ )بشكل أكثر وضوحاً.
- هذه الميزة تعادل **التعليقات التوضيحية** في لغات مثل **Python** و **C#**.
- يمكن استخدامها لتحديد خصائص إضافية على الكائنات أو دوال معينة.

مثال على استخدام Attributes :

```php
<?php

// تأكد من استيراد الفئات اللازمة من إطار Symfony.
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

// تعريف الكلاس HomeController مع السمة Route.
class HomeController
{
    // تعريف دالة تنفذ عندما يزور المستخدم '/home'.
    #[Route('/home')]
    public function index(): Response
    {
        // هنا، يمكنك إضافة منطق العمل مثل استرجاع بيانات من قاعدة البيانات أو معالجة أخرى.
        return new Response('مرحبًا بك في الصفحة الرئيسية!');
    }
}

```

الميزة توفر بديلاً أكثر تنظيماً ومرونة عن التعليقات التوضيحية.

### 2. Named Arguments (المعلمات المسماة):

- **المعلمات المسماة** تسمح بتمرير المعلمات إلى الدوال باستخدام أسمائها بدلاً من ترتيبها.
- هذه الميزة تسهل القراءة والصيانة، خاصة عندما تحتوي الدوال على العديد من المعلمات.

مثال على Named Arguments :

```php
<?php

// تعريف دالة greet التي تأخذ المعاملين name و age
function greet($name, $age) {
    // طباعة رسالة تحتوي على اسم الشخص وعمره
    echo "Hello $name, you are $age years old.";
}

// استدعاء الدالة مع استخدام التسمية (named arguments)
greet(age: 30, name: "John");

?>

```

### 3. Union Types (أنواع متعددة):

- **Union types تسمح بقبول أكثر من نوع واحد للمتغير أو المعامل.**
- **على سبيل المثال، يمكن أن يكون المتغير إما من النوع int أو string.**

```php
<?php

// تعريف الدالة foo إما يمكن أن تكون قيمة التي تأخذ int أو string
function foo(int|string $value): void {
    // طباعة القيمة التي تم تمريرها إلى الدالة
    echo $value;
}

// استدعاء الدالة مع قيمة من نوع int
foo(42);  // 42 ستقوم بطباعة

echo "<br>";  // إضافة فاصل بين النتيجتين

// استدعاء الدالة مع قيمة من نوع string
foo("Hello, World!");  // ستقوم بطباعة "Hello, World!"

?>
```

### 4. Constructor Property Promotion (ترقية الخصائص في المُنشئ):

- **ترقية الخصائص في المُنشئ** هي ميزة تسمح بترقية الخصائص مباشرة في مُنشئ الكلاس.
- هذا يقلل من الحاجة لكتابة الكود الزائد ويجعل الكود أكثر نظافة.

مثال على Constructor Property Promotion :

```php
<?php

// تعريف الكلاس Point
class Point {
    // تعريف الخصائص x و y كمتغيرات عامة (public)
    public int $x;
    public int $y;

    // تعريف دالة الإنشاء (constructor) لتهيئة القيم
    public function __construct(int $x, int $y) {
        $this->x = $x;
        $this->y = $y;
    }

    // إضافة دالة لعرض إحداثيات النقطة
    public function display(): void {
        echo "Point: ($this->x, $this->y)";
    }
}

// إنشاء كائن جديد من الكلاس Point مع إعطاء القيم لـ x و y
$point = new Point(10, 20);

// عرض الإحداثيات
$point->display();

?>
```

## Match Expressions.5 (تعابير التوافق) :

- **Match expressions هي تحسين للـ switchوتسمح بالمقارنة مع أنواع البيانات بشكل أكثر مرونة.**
- **يمكن أن تُعيد match قيمة مباشرة وتدعم التحقق من القيم بدون الحاجة إلى breaK**

مثال على Match Expressions

Lab3 > in.php

```php
<?php

// قيمة statusCode التي سيتم التحقق منها
$statusCode = 404;

// استخدام match للتحقق من قيمة statusCode
$result = match ($statusCode) {
    200 => 'OK',                // 200 إذا كانت القيمة
    404 => 'Not Found',         // 404 إذا كانت القيمة
    500 => 'Server Error',      // 500 إذا كانت القيمة
    default => 'Unknown',       // إذا كانت القيمة غير أي من القيم المحددة
};

// طباعة النتيجة
echo "Status: $result";

?>
```

## Weak Maps.6 (خرائط ضعيفة ) :

**Weak Maps هي هيكل بيانات جديد يسمح بتخزين الكائنات باستخدام** إشارة ضعيفة.

**هذه الخرائط لا تمنع جمع القمامة للكائنات التي تم تخزينها، مما يساعد في تحسين استخدام الذاكرة.**

مثال على Weak Maps :

Lab3 > in.php

```php
<?php

// قيمة statusCode التي سيتم التحقق منها
$statusCode = 404;

// استخدام match للتحقق من قيمة statusCode
$result = match ($statusCode) {
    200 => 'OK',                // 200 إذا كانت القيمة
    404 => 'Not Found',         // 404 إذا كانت القيمة
    500 => 'Server Error',      // 500 إذا كانت القيمة
    default => 'Unknown',       // إذا كانت القيمة غير أي من القيم المحددة
};

```

## 3. تحسينات في معالجة الأخطاء:

### 1.تحسينات في الـ Exceptions :

**تم تحسين** الاستثناءات **(Exceptions) في PHP 8 بحيث أصبحت أكثر وضوحاً في رسائل الخطأ.**

أضافة خاصية previous إلى **Exception** لتمرير استثناءات سابقة، مما يسهل تتبع الأخطاء.

### 2.Error Handling Enhancements :

تم تحسين آلية التعامل مع الأخطاء في PHP 8 حيث يتم تقديم رسائل أكثر تفصيلًا عند حدوث أخطاء.

أصبحت الأخطاء التي تتعلق بـ **TypeError** و **ValueError** أكثر وضوحاً في تعريف السبب.

## 4.التغييرات في الوظائف الحالية :

### 1.الإزالة من بعض الدوال والخصائص القديمة:

- تم إزالة بعض الدوال والخصائص التي كانت تعتبر قديمة أو غير آمنة.
- create_function() تم إلغاؤها لأنها كانت تؤدي إلى ثغرات أمان .

### 2.التغييرات على التوثيق:

- تم تحسين وتوضيح التوثيق على العديد من الدوال.

### 3.تحسينات في الدوال المتقدمة:

- تم تحسين بعض الدوال الخاصة بـ **date-time** و **strings** وأدوات التعامل مع **الملفات**.

## 5.المزايا الجديدة في التعامل مع أنواع البيانات :

### 1. تحسينات في التعامل مع الأنواع:

- تم إدخال تحسينات على طريقة التعامل مع الأنواع في PHP ، بما في ذلك تحسين **التحويل التلقائي للأنواع** (type coercion) .
- أصبحت **الأنواع الصارمة** أكثر توافقاً مع العمليات الحسابية، ما يجعل البرمجة أكثر أماناً ودقة.

### 2.Type System:

**PHP 8** يقدم دعمًا أوسع لأنظمة الأنواع مثل:

**1.Nullable Types**

**2. Return Type Coercion**

## 6.دعم أفضل لـ Static Analysis Tools :

- PHP 8 يوفر دعمًا أفضل للأدوات التي تقوم بتحليل الكود بشكل ثابت (مثل **PHPStan** و **Psalm** )وهذا يساعد في تحديد الأخطاء بشكل مبكر أثناء عملية التطوير.

## 7.التوافق مع الإصدارات السابقة :

- رغم التحديثات الكبيرة في PHP 8 ، فقد تم الحفاظ على التوافق مع الإصدارات السابقة لأغلب الأكواد.
- إلا أن بعض المميزات قد تم تعديلها أو إيقاف دعمها، لذا من المهم للمطورين القيام بفحص الكود الخاص بهم عند الانتقال من PHP 7 إلى PHP 8 .

# ماهو الفرق بين PHP8 والنسخ الذي قبلها

- الفرق بين **PHP 8** والإصدارات السابقة (مثل **PHP 7.4** و **PHP 7.x** )هو فرق كبير في العديد من الجوانب مثل الأداء، والميزات الجديدة، وتحسينات لغة البرمجة بشكل عام. سنقوم بمقارنة أبرز التغييرات والتحسينات التي تم تقديمها في **PHP 8** مقارنة بالإصدارات التي قبله، مثل **PHP 7.4**و **PHP 7.x**.

## 1.تحسين الأداء:

- PHP 7.x:
- قدم PHP 7 تحسينات كبيرة في الأداء مقارنة بـ PHP 5.x بفضل Zend Engine 3.0.
- كان الفرق بين PHP 7 و PHP 5 كبيرًا جدًا في سرعة التنفيذ وكفاءة الذاكرة.
- PHP 8 يقدم تحسينات أكبر في الأداء بفضل JIT (Just In Time Compilation).
- الـ JIT يُعد أحد أكبر التحسينات في PHP 8، حيث يتم ترجمة الكود إلى تعليمات الآلة في وقت التنفيذ، مما يساعد في تسريع أداء التطبيقات خصوصًا في العمليات الحسابية الثقيلة مثل التطبيقات العلمية أو الألعاب.
- مع PHP 8، يلاحظ بعض المطورين تسريعًا في تنفيذ الأكواد بشكل عام، خاصة في التطبيقات المعقدة.

## 2.المزايا الجديدة فيPHP 8 :

- PHP 7.x:
- قدم تحسينات مثل Type Declarations (إعلانات الأنواع) و Return Type Declarations، مما جعل الكود أكثر وضوحًا ودقة.
- دعم الإعلانات الصارمة للأنواع (Strict Types).
- تحسين في Error Handling حيث تم إدخال Error exceptions بدلًا من Warnings و Notices.
- PHP 8:
- JIT Compiler: كما ذكرنا، أصبح PHP 8 يستخدم JIT لزيادة السرعة.
- Attributes (السمات): تم تقديم Attributes كبديل للتعليقات التوضيحية (annotations)، مما يتيح إضافة بيانات ميتا أكثر مرونة ووضوحًا على الكائنات أو الدوال.
- Named Arguments: أصبحت PHP 8 تدعم المعلمات المسماة، مما يتيح للمطورين تمرير القيم إلى دوال باستخدام الأسماء بدلاً من ترتيب المعلمات.
- Match Expressions: تقديم match كبديل أكثر قوة ودقة لـ switch، يدعم التحقق من الأنواع وأيضًا عدم الحاجة إلى استخدام break.
- Constructor Property Promotion: تحسين كبير لتقليل الكود المتكرر عند تعريف الكائنات من خلال السماح بترقية الخصائص مباشرة في المُنشئ.
- Union Types: دعم الأنواع المتعددة (مثل int|string)، حيث يمكن للمتغير قبول أكثر من نوع واحد.
- Weak Maps: تم تقديم Weak Maps التي تسمح بتخزين الكائنات باستخدام إشارة ضعيفة، مما يساعد في إدارة الذاكرة بشكل أفضل.

## 3. التغييرات في التعامل مع الأخطاء

- PHP 7.x:
- في PHP 7 تم تقديم Throwable interface مما يعني أن Errors و Exceptions أصبحوا يعملون بشكل متشابه، مع وجود TypeError و ParseError التي تعالج الأخطاء البرمجية.
- PHP 8:
- تم تحسين Error Handling بشكل كبير في PHP 8، حيث أصبحت الأخطاء تظهر رسائل أكثر وضوحًا وتفصيلًا.
- تم تحسين دعم الـ Exceptions حيث يمكن الآن تتبع الأخطاء عبر Previous Exception.
- تم تقديم ValueError و TypeError بشكل أفضل للتعامل مع الأخطاء المتعلقة بالأنواع.

## 4. التغييرات في إدارة الأنواع

- PHP 7.x:
- دعم قوي للأنواع عبر Type Hinting و Return Type Declarations، كما كان يسمح بإجبار الدوال على قبول أنواع معينة من المدخلات.
- PHP 8:
- دعم الأنواع في PHP 8 أصبح أكثر مرونة ودقة.
- تم تقديم Union Types حيث يمكن أن تكون المتغيرات أو الدوال تقبل أنواع متعددة، على سبيل المثال int|string.
- تم تحسين التعامل مع Nullable Types (أنواع قابلة لأن تكون فارغة)، مما يتيح التعامل مع القيم الفارغة بشكل أفضل.

## 5. دعم الأدوات الخارجية وتحليل الكود

- PHP 7.x:
- PHP 7 حسّن من تكامل اللغة مع الأدوات الخارجية مثل PHPStan و Psalm.
- PHP 8:
- في PHP 8، تمت إضافة دعم أكبر لتحليل الكود الثابت باستخدام أدوات مثل PHPStan و Psalm، مع تحسينات في التوثيق ودعم الأنواع. هذا يسهل اكتشاف الأخطاء البرمجية قبل وقت التنفيذ.

## 6. الإضافات والتحسينات الأخرى

- PHP 7.x:
- PHP 7.4 شهد بعض التحسينات مثل Typed Properties (الخصائص المحددة بأنواع) و Preloading للملفات، مما ساعد في تقليل وقت تحميل الصفحة.
- PHP 8:
- PHP 8 يضيف تحسينات مثل Named Arguments التي تسهل التعامل مع الدوال التي تحتوي على العديد من المعلمات.

- تقديم Match Expressions كطريقة جديدة ومُحسنة للتعامل مع المقارنات بدلاً من switch.
- Constructor Property Promotion جعل الكتابة أسهل وأسرع في إنشاء الكائنات.
- التحسينات على إدارة الذاكرة باستخدام Weak Maps.

## 7. الإزالة أو التغيير في الخصائص القديمة

- PHP 7.x:
- PHP 7 كان يقدم بعض الوظائف التي تم تحسينها أو إيقاف استخدامها تدريجيًا مثل create_function().
- PHP 8:
- PHP 8 تخلص من العديد من الدوال التي كانت قديمة أو خطرة، مثل create_function() و each()، كما تم إيقاف دعم بعض الخصائص القديمة التي كانت تعتبر غير آمنة أو غير فعالة.
- كما أن PHP 8 جعل استخدام Deprecated Features أكثر وضوحًا.

## 8. دعم SSL وتوثيق الاتصال

- PHP 7.x:
- في PHP 7 تم تحسين التعامل مع SSL وأدوات cryptography.
- PHP 8:
- في PHP 8، استمر التحسين في استخدام SSL وTLS، وتم تحسين دعم أمان الاتصال بشكل عام.

خاتمة:

تعتبر PHP 8 نقلة نوعية في تاريخ لغة PHP. من خلال تقديم JIT، وتحسينات كبيرة في الأداء، وميزات جديدة مثل Attributes و Union Types و Match Expressions، تقدم PHP 8 أدوات قوية للمطورين لتحسين الأداء وكتابة كود أكثر نظافة وقوة. هذه التحسينات لا تقتصر فقط على تسريع الأداء، بل تعزز تجربة البرمجة بشكل عام وتساعد المطورين على بناء تطبيقات أكثر كفاءة وقوة.